Trabajo práctico 1 – Inteligencia artificial
UNaB 2023 cuat. 2

G. Sebastián Pedersen

sebastian.pedersen@unab.edu.ar

Lunes 11 de Septiembre de 2023

1. Ponderación de este TP respecto a la nota final: 30 %.
2. Entregar las resoluciones de las partes teóricas en un único archivo `pdf` con nombre `tp1_teo_IA_apellido_nombre.pdf`.
3. Entregar las resoluciones de las partes de programación en un único archivo `zip` con nombre `tp1_prog_IA_apellido_nombre.zip`. El archivo `zip` debe contener únicamente los scripts y salidas de los mismos.
4. Entregar por el campus.
5. Fecha y hora límite de entrega: Dom 15/Oct/2023 23:59 hs. (GMT-3).
6. El TP es de entrega individual.
7. **El cumplimiento de las pautas de entrega es parte de la resolución del TP.**

1. **Regresión logística: estabilidad del entrenamiento**

   En este problema, profundizaremos en el funcionamiento de la regresión logística. El objetivo es desarrollar habilidades de depuración de algoritmos de aprendizaje automático (que puede ser muy diferente del de software en general).

   Se proporciona una implementación de regresión logística en `src/p01_lr.py`, y dos conjuntos de datos etiquetados A y B en `data/ds1_a.txt` y `data/ds1_b.txt`.

   No modifique el código de entrenamiento de regresión logística para este problema. Primero, ejecute el código de regresión logística dado para entrenar dos modelos diferentes en A y B. Puede ejecutar el código simplemente corriendo `python p01_lr.py` en el directorio `src`.

   a) ¿Cuál es la diferencia más notable al entrenar el modelo de regresión logística en los conjuntos de datos A y B?

   b) Investigue por qué el procedimiento de entrenamiento se comporta de manera inesperada en el conjunto de datos B, pero no en A. Proporcione evidencia sólida (matemática, código, gráficos, etc.) para corroborar su hipótesis. Recuerde, debe justificar por qué su explicación *no* se aplica a A.

   Comentario: el problema no es de redondeo numérico o un error de under/overflow.

   c) Para cada una de las siguientes posibles modificaciones, indique si conduciría o no a que el algoritmo de entrenamiento proporcionado converja en el conjunto de datos B. Justifique sus respuestas.

   i) Usar una tasa de aprendizaje constante diferente.

   ii) Reducir la tasa de aprendizaje con el tiempo (por ejemplo, escalar la tasa de aprendizaje inicial en $1/t^2$, donde t es el número de iteraciones de descenso por gradiente hasta el momento).

   iii) Escalado lineal de los datos de entrada.

   iv) Agregar un término de regularización $|\theta|_2^2$ a la función de costo (también llamada de pérdida).

   v) Agregar ruido gaussiano de media cero a los datos o a las etiquetas.

   d) ¿Las SVM, que usan la pérdida de hinge, son vulnerables a conjuntos de datos como el B? ¿Por qué o por qué no? Dar una justificación informal.

   Comentario: buscar en SVM la diferencia entre margen funcional y margen geométrico.

2. **Calibración de modelos.**

En este problema intentaremos entender la salida $h_\theta(x)$ de la función de hipótesis de una regresión logística, en particular por qué podríamos tratar la salida como una probabilidad (además del hecho que la función sigmoidea asegura que $h_\theta(x)$ siempre se encuentra en el intervalo $(0, 1)$).

Cuando las probabilidades generadas por un modelo coinciden con la observación empírica, se dice que el modelo está *bien calibrado* (o es confiable). Por ejemplo, si consideramos un conjunto de ejemplos $x^{(i)}$ para los cuales $h_\theta(x^{(i)}) \simeq 0.7$, alrededor del 70 % de esos ejemplos deberían tener etiquetas positivas. En un modelo bien calibrado, esta propiedad se cumplirá para cada valor de probabilidad.

La regresión logística tiende a generar probabilidades bien calibradas (esto a menudo no es cierto con otros clasificadores como Naive Bayes o SVM). Vamos a profundizar un poco más para entender por qué este es el caso, y entender que la estructura de la función de pérdida explica esta propiedad.

Supongamos que tenemos un conjunto de entrenamiento $\{x^{(i)}, y^{(i)}\}_{i=1}^m$ con $x^{(i)} \in \mathbb{R}^{n+1}$ e $y^{(i)} \in \{0, 1\}$. Supongamos que tiene un término independiente $x_0^{(i)} = 1$ para todo i. Sea $\theta \in \mathbb{R}^{n+1}$ los parámetros de máxima verosimilitud aprendidos después de entrenar un modelo de regresión logística. Para que el modelo sea considerado bien calibrado, dado cualquier rango de probabilidades (a, b) tal que $0 \leq a < b \leq 1$, y datos de entrenamiento $x^{(i)}$ donde los resultados del modelo $h_\theta(x^{(i)})$ caen en el rango (a, b), la fracción de positivos en ese conjunto de datos debe ser igual al promedio de los resultados del modelo para esos datos. Es decir debe cumplirse la siguiente propiedad:

$$\frac{\sum\limits_{i \in I_{a,b}} P\left(y^{(i)} = 1 | x^{(i)}, \theta\right)}{|\{i \in I_{a,b}\}|} = \frac{\sum\limits_{i \in I_{a,b}} \mathbb{I}\{y^{(i)} = 1\}}{|\{i \in I_{a,b}\}|}$$

donde $P(y = 1|x, \theta) = h_\theta(x)$ es la sigmoide, y donde $|S|$ denota el tamaño del conjunto S.

a) Demuestre que la propiedad anterior se cumple para el modelo de regresión logística descripto, sobre el rango $(a, b) = (0, 1)$.

Sugerencia: use el hecho de que se incluyó un término indepentiente (también llamado de sesgo).

b) Si tenemos un modelo de clasificación binaria que está perfectamente calibrado, es decir, la propiedad que acabamos de demostrar se cumple para cualquier $(a, b) \subseteq [0, 1]$, ¿implica esto necesariamente que el modelo logra una precisión perfecta? ¿Es la recíproca necesariamente cierta? Justifique sus respuestas.

c) Discuta qué efecto tiene en la calibración del modelo incluir regularización L_2 en la función objetivo de la regresión logística.

Observación: consideramos el rango $(a, b) = (0, 1)$. Esta es el único rango para el cual está garantizado que la regresión logística está calibrada en el conjunto de entrenamiento. Cuando el modelo GLM es válido todos los rangos $(a, b) \subseteq [0, 1]$ están bien calibrados. Además, cuando el conjunto de entrenamiento y test son de la misma distribución y cuando el modelo no tiene sobreajuste o subajuste, la regresión logística tiende a estar bien calibrada en datos de test *no vistos aún*. Esto hace que la regresión logística sea un modelo muy popular en la práctica, especialmente cuando estamos interesados en el nivel de incertidumbre de la salida.

3. **Interpretación bayesiana de la regularización.**

En la estadística bayesiana, casi todas las cantidades son variables aleatorias, que pueden ser observadas o no. Por ejemplo, los parámetros θ son generalmente variables aleatorias no observadas, y los datos x e y son variables aleatorias observadas. La distribución conjunta de todas las variables aleatorias también se denomina modelo (por ejemplo, $p(x, y, \theta)$). Toda cantidad desconocida se puede estimar condicionando el modelo a todas las cantidades observadas. Tal distribución condicional sobre las variables aleatorias no observadas, condicionadas a las variables aleatorias observadas, se denomina distribución posterior. Por ejemplo, $p(\theta|x, y)$ es la distribución posterior en el contexto del aprendizaje automático. Una consecuencia de este enfoque es que estamos obligados a dotar a los parámetros con una distribución prior $p(\theta)$. Las probabilidades priors deben ser asignadas antes de que veamos los datos: necesitan capturar nuestras creencias previas sobre cuáles podrían ser los parámetros del modelo antes de observar cualquier evidencia, y debe ser una opinión subjetiva de la persona que construye el modelo.

En la interpretación bayesiana más pura, estamos obligados a mantener la distribución posterior sobre los parámetros todo el camino hasta llegar a la predicción, para (justamente) llegar a la distribución de predicción posterior, y la predicción final será el valor esperado de la distribución predictiva posterior. Sin embargo, en la mayoría de las situaciones, esto es computacionalmente muy costoso, y nos conformamos con algo que es menos puro (en el sentido bayesiano).

El compromiso es estimar un valor puntual de los parámetros (en lugar de la distribución completa) que es la moda de la distribución posterior. Estimar la moda de la distribución posterior también se denomina estimación máxima a posteriori (MAP). Esto es

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|x, y)$$

Compare esto con la estimación de máxima verosimilitud (MLE) que hemos visto anteriormente:

$$\theta_{MLE} = \underset{\theta}{\operatorname{argmax}}\, p(y|x, \theta)$$

En este problema, exploramos la conexión entre la estimación MAP, y técnicas de regularización populares que se aplican con la estimación MLE. En particular, mostraremos cómo la elección de la distribución prior sobre θ (por ejemplo gaussiana) es equivalente a diferentes tipos de regularización (por ejemplo regularización L_2 o L_1). Para mostrar esto, iremos procediendo paso a paso:

a) Demuestre que $\theta_{MAP} = \underset{\theta}{\operatorname{argmax}}\, p(y|x, \theta)p(\theta)$ si suponemos que $p(\theta) = p(\theta|x)$. Esta suposición será válida para modelos como la regresión lineal donde la entrada x no está explícitamente modelada por θ. (Tenga en cuenta que esto significa que x y θ son marginalmente independientes, pero no condicionalmente independientes cuando y es dado).

b) Recuerde que la regularización L_2 penaliza la norma L_2 de los parámetros mientras minimiza la pérdida o costo (es decir, NLL en el caso de modelos probabilísticos). Ahora mostrará que la estimación MAP con un prior sobre θ gaussiano de media cero, específicamente $\theta \sim N(\theta, \eta^2 I)$, es equivalente a aplicar regularización L_2 con estimación MLE. Específicamente, muestre que:

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmin}} \left(-\log \left[p(y|x, \theta) \right] + \lambda |\theta|_2^2 \right)$$

Además, ¿cuál es el valor de λ?

c) Ahora considere un modelo específico, un modelo de regresión lineal dado por $y = \theta^t x + \epsilon$ donde $\epsilon \sim N(\theta, \sigma^2)$. Como antes, suponga una prior gaussiana $\theta \sim N(\theta, \eta^2 I)$. Sea X la matriz de todas las entradas de los datos de entrenamiento donde cada fila es un dato, e $\vec{y}$ es el vector columna de todas las salidas de los datos.

Hallar una expresión cerrada para θ_{MAP}.

d) A continuación, considere la distribución de Laplace, cuya densidad viene dada por

$$f_{\mathcal{L}}(z|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|z-\mu|}{b}\right)$$

Como antes, considere un modelo de regresión lineal dado por $y = \theta^t x + \epsilon$ donde $\epsilon \sim N(\theta, \sigma^2)$. Suponga un prior de Laplace $\theta \sim \mathcal{L}(0, bI)$. Demostrar que θ_{MAP} en este caso es equivalente a la solución de regresión lineal con regularización L_1, cuya pérdida viene dada por:

$$J(\theta) = |X\theta - \vec{y}|_2^2 + \gamma|\theta|_1$$

Además, ¿cuál es el valor de γ?

Observación: una solución con forma cerrada para el problema de regresión lineal con regularización L_1 no existe. Para optimizar esto, usamos el descenso por gradiente con una inicialización aleatoria y lo resolvemos numéricamente.

Observación: la regresión lineal con regularización L_2 también se denomina comúnmente regresión Ridge, y cuando se emplea la regularización L_1 se denomina comúnmente regresión de Lasso. Estas regularizaciones se puede aplicar a cualquier modelo lineal generalizado como se indicó anteriormente (reemplazando $\log p(y|x, \theta)$ con la densidad de probabilidad apropiada). Las técnicas de regularización del tipo anterior también se denominan de decaimiento de pesos, y de contracción. Los priors gassuiano y de Laplace fomentan a que los valores de los parámetros esten más cerca de su media (es decir, cero), lo que da como resultado un efecto de contracción.

Observación: se sabe que la regresión de Lasso (es decir, la regularización L_1) da como resultado parámetros ralos, es decir donde la mayoría de los valores de los parámetros son cero, y solo algunos de ellos son distintos de cero.

4. **K-means para compresión.**

   En este problema, aplicaremos el algoritmo K-means a la compresión de imágenes con pérdida, reduciendo el número de colores utilizados en una imagen.

   Usaremos los archivos `data/peppers-small.tiff` y `data/peppers-large.tiff`.

   El archivo `peppers-large.tiff` contiene una imagen de 512x512 de pimientos representados en color de 24 bits. Esto significa que, para cada uno de los 262144 píxeles de la imagen, hay tres números de 8 bits (cada uno que va de 0 a 255) que representan los valores de intensidad de rojo, verde y azul para ese píxel. Por lo tanto, la representación directa de esta imagen ocupa alrededor de $262144 \times 3 = 786432$ bytes (un byte siendo 8 bits). Para comprimir la imagen, usaremos K-means para reducir la imagen a $k = 16$ colores. Más concretamente, cada píxel de la imagen se considera un punto en el espacio tridimensional (r, g, b). Para comprimir la imagen, agruparemos estos puntos en 16 grupos (los 16 colores), y reemplazaremos cada píxel con el centroide del cluster más cercano.

   Sigue las instrucciones de más abajo. Tenga en cuenta que algunas de estas operaciones pueden tardar un tiempo (¡varios minutos incluso en una computadora rápida!)

   a) [**Programación**] **Implementación de compresión por K-Means**. Desde el directorio `data`, abra un intérprete interactivo de Python, ejecute:

   `from␣matplotlib.image␣import␣read;␣import␣matplotlib.pyplot␣as␣plt`

   y ejecuta `A␣=␣imread('peppers-large.tiff')`. Ahora `A` es una "matriz tridimensional", y `A[:,:,0]`, `A[:,:,1]` y `A[:,:,2]` son matrices de 512x512 que contiene respectivamente los valores de rojo, verde y azul para cada píxel. Ejecute `plt.imshow(A);␣plt.show()` para mostrar la imagen.

   Dado que la imagen grande tiene 262144 píxeles y tardaría un tiempo en agruparse en clusters, en su lugar trabajaremos la anterior vectorización en la imagen más pequeña. Repita lo anterior con `peppers-small.tiff` tratando los valores de cada píxel (r, g, b) como un elemento de $\mathbb{R}^3$. Ejecute K-means con 16 clusters, iterando preferentemente hasta la convergencia, pero en ningún caso por menos de 30 iteraciones. Para la inicialización, establezca cada centroide como los valores (r, g, b) de un píxel elegido al azar en la imagen.

   Tome la matriz `A` de `peppers-large.tiff` y reemplace los valores (r, g, b) de cada píxel con el valor del centroide del clúster más cercano. Muestre la nueva imagen y compárela visualmente a la imagen original. Incluya en su resolución todo su código y una copia de su imagen comprimida.

   Implemente todo su código en un archivo de nombre `p04_kmeans.py`.

   b) **Factor de compresión**. Si representamos la imagen con estos (16) colores reducidos, ¿por (aprox.) qué factor estamos comprimiendo la imagen?